Athens Institute for Education and Research
ATINER

ATINER's Conference Paper Series
COM2012-0261

A Partitioned Stochastic Search Algorithm: Application to Multi-unit Winner Determination Problem in Combinatorial Auction

Raj Jog Singh
IBM India Private Limited
India

Anup Kumar Sen
Professor
Indian Institute of Management Calcutta
India

Uttam Kumar Sarkar
Professor
Indian Institute of Management Calcutta

India

Athens Institute for Education and Research
8 Valaoritou Street, Kolonaki, 10671 Athens, Greece
Tel: + 30 210 3634210 Fax: + 30 210 3634209
Email: info@atiner.gr URL: www.atiner.gr
URL Conference Papers Series: www.atiner.gr/papers.htm

Printed in Athens, Greece by the Athens Institute for Education and Research.

ISSN **2241-2891**
19/09/2012

An Introduction to ATINER's Conference Paper Series

ATINER started to publish this conference papers series in 2012. It includes only the papers submitted for publication after they were presented at one of the conferences organized by our Institute every year. The papers published in the series have not been refereed and are published as they were submitted by the author. The series serves two purposes. First, we want to disseminate the information as fast as possible. Second, by doing so, the authors can receive comments useful to revise their papers before they are considered for publication in one of ATINER's books, following our standard procedures of a blind review.

Dr. Gregory T. Papanikos
President
Athens Institute for Education and Research

This paper should be cited as follows:

Singh, R.J., Sen, A.K. and Sarkar, U.K. (2012) **"A Partitioned Stochastic Search Algorithm: Application to Multi-unit Winner Determination Problem in Combinatorial Auction"** Athens: ATINER'S Conference Paper Series, No: COM2012-0261.

A Private Hospital Example for the Analysis of Unit Costs in Hospital Enterprises

Raj Jog Singh
IBM India Private Limited
India

Anup Kumar Sen
Professor
Indian Institute of Management Calcutta
India

Uttam Kumar Sarkar
Professor
Indian Institute of Management Calcutta
India

Abstract

With the penetration of the Internet and the ease of carrying out an auction, the electronic auction market has already become huge and is expanding rapidly. The paper presents a novel stochastic partition-based search technique to solve the winner determination problem associated with the multi-unit combinatorial auction. The algorithm uses an innovative divide and conquer formulation in the search framework so that solutions to sub-problems remain additive. The proposed technique can be used to find both optimal as well as near optimal solutions. Experimental results with the benchmark problem suites using uniform and decay distributions are presented. The results exhibit saving in computation can be as high as 99% while compromising less than 1% of the revenue.

Keywords: Combinatorial auction, winner determination problem, electronic commerce, branch and bound, NP-hard, local search, heuristic search

Contact Information of Corresponding author:

Introduction

Auction theory and methodologies have drawn a strong attention of researchers since the seminal work of Vickrey [Vickrey 1961] and several auction models exist (see [Wilson 1969], [Milgrom and Weber 1982], [Clarke 1971], [Groves 1973]). With the penetration of the Internet and the ease of carrying out an auction by both the bidders and the auctioneer, the electronic auction market is already huge and is expanding rapidly. With faster access to necessary information, the bidders and auctioneers can now apply sophisticated optimization procedures which would be infeasible in the domain of manually conducted auctions.

In Combinatorial Auctions, multiple goods (items) are available for auction simultaneously, and bidders bid for combinations of goods called bundles. Often the value of a good to a potential buyer may depend upon what other goods he wins in the auction. That is, there exists *complimentarity* between two goods, say gB_{1B} and gB_{2B}, to bidder j if $UB_{j\,B}\{ gB_{1B}\} + UB_{j\,B}\{ gB_{2B}\} < UB_{j\,B}\{ gB_{1B}, gB_{2B}\}$ where UB_{jB} $\{SB_{jB}\}$ is the utility to bidder j of acquiring the set of goods SB_{jB}. Traditional auction mechanisms would require a need for look-ahead on part of the bidder in such scenarios as well as provisions to deal with inefficiencies arising from uncertainty. In such a scenario, bidders would prefer to bid for combination of items referred to as bundles of goods. Such an auction mechanism has many real life applications such as allocation of railroad, auction of adjacent pieces of real estate, auction of airport landing slots, and distributed job shop scheduling.

The *winner determination problem (WDP) of combinatorial auction* to find the winning bids has thus become immensely important. The auctioneer attempts to maximize his auction revenue under constraints imposed by the availability of items and known information about the bids. The winner determination problem is hard computationally. In a conventional single unit WDP, each item for auction has only unit and consequently, at most one bidder may eventually win this item. In contrast, multi-unit combinatorial auctions involve circumstances where sellers have to sell a number of identical units of items and buyers may also be interested to buy more than one unit of items through a single bid. Selling all the items of a store in case the store winds up, buying components of personal computers for assembling the PCs in a wholesale market where prices are discovered through auctions are typical examples of such auctions.

The present research deals with multi-unit combinatorial auctions where the auctioneer has a set of distinguishable items with a specific number of identical units for each of these items. So, the bidder is concerned only with whether he gets the requisite number of units of the items that interest him and not about which specific units of an item he gets. The bidders place their bids on the combinations they desire stating exactly the number of units of the items they want and the price they are willing to pay for it. Unlike the single unit case, here we can have bids of identical composition placed in the auction with

a possibility of more than one of them getting allocated or winning as long as they together are not asking for more units than what the auctioneer has.

In this paper we propose a stochastic partition based divide and conquer strategy in a depth-first branch and bound search framework, called *multi-partitioning* search, which significantly reduces the computational overhead in solving optimal and near-optimal solutions to large problem instances of multi-unit winner determination problem. Extensive experiments with problem suites from uniform and decay distributions exhibit savings in computation time as high as 99% while compromising less than 1% of the revenue. The algorithm reduced the CPU time running into hours and days for very large problem instances into a few seconds with negligibly small loss in revenue.

The rest of the paper is organized as follows. Literature survey is presented in Section 2. The multi-unit winner determination problem addressed in this paper is defined in Section 3. Section 4 presents LSMU(β), a parameterized algorithm, that incorporates multi-partitioning search concept. The functioning of the algorithm is explained with an example. The experimental setup for evaluating the performance of the algorithm is elaborated in Section 5 and the findings are given in Section 6. Section 7 draws concluding remarks.

Literature Survey

The single-unit version of the problem has been well studied and solutions were attempted in various ways ([Cramton *et al.* 2006], [Adomavicius and Gupta 2005]). Some imposed severe restrictions on bids so that the problem can be solved faster ([Rothkopf *et al.* 1998], [Muller 2006]). Heuristic search algorithms like CASS [Fujishima *et al.* 1999] and CABOB ([Sandholm *et al.* 2001], [Sandholm 2002], [Sandholm and Suri 2003], [Sandholm 2006]) have high overhead of computation for solving large problem instances. Stochastic search method that aims at finding near-optimal solutions has also been tried ([Hoos and Boutilier 2000], [Hoos and Stutzle 2004], [Singh and Sen 2005]). Iterative versions of single-unit combinatorial auctions have been studied in ([Ausubel and Milgrom 2002], [Parkes 1999, 2006]).

The multi-unit winner determination problem (MWDP) is NP-hard [Garey and Johnson 1979] - even its special case variant where each item can have only one unit is known to be NP-hard ([Rothkopf *et al.* 1998], [Lehman and Sandholm 2006]). The decision version remains NP-complete even if we restrict instances where every bid has a value equal to 1, and every bidder bids only on subsets of size at most 3 [Rothkopf *et al.* 1998]. Consequently, large instances of MWDP, are known to be intractable due to combinatorial explosion and it has been experienced in the findings of CAMUS [Kevin *et al.* 2000b].

In contrast to the methods known for solving single-unit WDP, available literature to solve multi-unit MWDP are not many in number. Kevin et al. [2000b] suggested CAMUS, a generalization of CASS, for solving optimally the MWDP associated with the multi-unit scenario. The algorithm has the computational overhead of branch-and-bound search, and in its given form, might fail to output optimal solutions [Ghebreamiak *et al.* 2002]. A branch and bound formulation [Gonen and Lehman 2000] and a LP formulation [Gonen and Lehman 2002] were proposed but combinatorial explosion reduces its utility for solving large instances. Xia and Whinston [2005] have suggested a mechanism for transforming the combinatorial double auction to an equivalent single-sided auction. A decision support approach has been proposed in [Koksalan *et al.* 2009].

Multi-unit Winner Determination Problem (MWDP)

In the multi unit combinatorial auction, the auctioneer has a set of distinguishable items with a specific quantity of units for each of these items. The bidders place their bids on the combinations they desire stating exactly the number of units of the items they want and the price they are willing to pay for it.

- A set of items/goods to auction $M = \{1, 2, 3,\ldots, m\}$.
- An integer $q(k)$ denoting the number of units of good k available on auction.
- A set of bids $B = \{ BB_{1B}, BB_{2B}, BB_{3B}, \ldots , BB_{nB}\}$ where a bid $BB_{jB} = (SB_{jB}, pB_{jB})$ where pB_{jB} is the price offer of bid BB_{jB} and $SB_{jB}=(\{ qB_{jB}(1), qB_{jB}(2), qB_{jB}(3), \ldots, qB_{jB}(m) \}$ where $qB_{jB}(k)$ = number of units of good k in bid BB_{jB}. If BB_{jB}requires no units of a particular good k then $qB_{jB}(k) = 0$. Each bid BB_{jB} is associated with a unique bidder i. A bidder may make several bids and eventually win zero or more bundle of items.

The multi-unit winner determination problem (MWDP) is to label each bid BB_{jB} as either winning ($x_j = 1$) or losing ($x_j = 0$) so as to maximize the auctioneer's revenue under the constraint that for any item k, the sum of units of k over all the winning bids does not exceed $q(k)$. Mathematically,

$$\text{Max} \sum_{j=1}^{n} p_j x_j \ \textit{such that}$$

$$\sum_{j=1}^{n} (q_j(k) \times x_j) \le q(k), \ \textit{for all } k, k = 1, 2, 3, ..., m \ \textit{and} \ x_j \in \{0,1\}$$

Unlike the single unit case, here we can have bids of identical composition placed in the auction with a possibility of more than one of them getting

allocated or winning as long as they together are not asking for more units than what the auctioneer has.

We now present LSMU(β), our multi partitioned local search approach.

Multi-partitioned Local Search Algorithm LSMU(β) for MWDP

The LSMU(β) is a parameterized algorithm that uses multi-partitioned search concept. Here β denotes the number of partitions of a feasible solution. The details of the algorithm are described below.

Multi-partitioned local search

In this proposed approach a single MWDP instance is split randomly into multiple *additive* smaller MWDP instances. Each of these smaller instances is optimally solved faster using depth-first branch and bound (DFBB) method. An *additive* decomposition ensures the resulting revenue would be the sum of the revenues obtained by solving the subproblems. The task of finding additive sub-problems becomes nontrivial because the splitting has to be such that the solutions obtained after solving each sub-problem can be directly combined and yet the availability constraint is not violated that is, no more than available number of units of each item gets sold by the auctioneer. The challenge of the approach therefore, lies in an effective mechanism to deal with the additive decomposition problem. Repeated trials of decomposition and the solutions of smaller sub-problems are made, and the overall best solution is chosen.

The parameter β is a positive integer that controls the granularity of partitions. It makes use of an initial feasible solution to the given MWDP, LSMU (β) partitions the problem into β compartments stochastically and solves each of these compartments by search techniques. When β = 1, LSMU(1) outputs optimal solutions. As β *increases* from 1 to higher values the cost of computation decreases while the quality of solution deteriorates. What makes LSMU(β) exciting and worth exploring is the nature of these reductions in computational time and solution quality that suggest suitable values of β for which very near optimal (say, one within a deviation of only 1% or less from the optimal revenue) solutions can be obtained with significant reduction (say, 95% or more) of computational time of obtaining the optimal solution.

Algorithm LSMU(β)

This section presents the algorithm for solving MWDP. We present a sketch of the algorithm below.

Algorithm LSMU(β)

Begin

Pre-process bids, reorder goods and allocate bids to bins to improve search;

Find the initial feasible solution containing η bids; this is the present solution;

Repeat

Randomly divide the η bids in the present solution into β baskets;

For each of the β baskets
Remove bids in the basket from the present solution;
Form a subproblem based on the bids removed;
Optimally solve the sub-problem;
Use the solution to find a new feasible (neighbourhood) solution by adding the bid values of the other ($\beta - 1$) baskets*;*
If improvement then update the present solution;
End For
Until T consecutive non-improving iterations*;*
End;

We now explain some of the steps below.

- *D*uring preprocessing, in LSMU(β) the singleton bids are processed using dynamic programming as is suggested in [Kevin, 2000b]. A singleton bid is a bundle consisting of a single item though there can be multiple units of it. After preprocessing, the singleton bids are removed from the problem and replaced by singleton vectors. A singleton vector for any item is the best possible collection of singleton bids for a given number of units of that item to be allocated. This helps to allocate a chunk of units of an item together.
- The goods are ordered following a heuristic to ensure there is more branching at the initial stages and less at the later stages of search. For each good *k*, $order_k$ is computed as $1/(numbids_k \times avgunit_k)$, where $numbids_k$ is the number of bids that requests good *k* and $avgunit_k$ is the average number of goods requested by these bids. The good with the lowest order is designated as the lowest ordered good.
- Bins of bids are formed to speed up the process of selecting feasible bids. [Fujishima, 1999]. Given any ordering of goods, there is one bin for each good and each bid belongs to the bin corresponding to its lowest numbered good. The bids inside a bin are ordered in non-increasing order of bid price per unit item.
- The initial feasible solution is found by including feasible bids from the set of bids arranged in non-increasing order of bid price per unit item until no further bid can be allocated.
- Randomly divide the bids in the present solution into β baskets each preferably containing $[\eta/\beta]$ bids.
- Every basket in the present solution creates a sub-problem. In forming a sub-problem corresponding to a basket, the bids in the basket are removed from the present solution. The associated units of items pertaining to these bids thus become free or unallocated and are ready to be auctioned by the set of bids that does not belong to all the other baskets. This form a separate winner determination sub-problem of a smaller size.
- The smaller sub-problem is optimally solved using DFBB search algorithm. At any node of the search tree, a feasible bid corresponding to lowest numbered unallocated item is chosen from the corresponding bin and added to the path. An upper bound is used in LSMU(β) to prune paths

based on revenue estimates. At any node during search, for every unallocated item, the highest ordered (in non-increasing order of bid price per unit item) feasible bid containing this item is selected and its contribution is added to obtain an upper bound.

- The allocation generated by solving the sub-problem is added to the bids from the remaining baskets giving a complete feasible solution and the revenue is computed. This gives a neighbourhood solution. Since the baskets may have bids containing one or more common items, the order in which search is conducted on the baskets will influence the resulting revenue for that iteration. That's the reason we evaluate the quality of the solution after considering every basket.

An Example

The example illustrates the performance of LSMU(β). For simplicity, we assume $\beta=2$**.** Consider an instance of three items = {1, 2, 3} to be auctioned with three available units of each item. The set of the bids in the format {(item, units): price} obtained after renumbering, preprocessing and their allocation into bins is given in Table 3.1. The first row shows the given set of bids before preprocessing. During preprocessing, the singleton vectors and corresponding bids for every item is created. For item 1, the singleton vectors are created as {(1,1):$20}, {(1,2):$30}, {(1,3):$45} and {(1,4):$55}, and the singleton bids are removed.

The initial feasible solution is the set of bids = {(1,1), (2,2):$70}, {(1,1):$20}$P^{\circ P}$, {(2,1),(3,2):$60}, {(3,1):$30} with revenue = $180. After bifurcating (since β=2) the bids in the initial solution into baskets their contents become as follows:

Table 1: *Bids and Bins of the Example instance*

Items	Set of bids = {(1,1):$20}, {(1,1):$10}, {(1,2):$25}, {(1,1), (2,2):$70}, {(2,2):$60}, {(2,2), (3,1):$75}, {(2,1),(3,2):$60}, {(3,1):$30} After treating singletons with dynamic programming: {(1,1):$20}$P^{\circ P}$, {(1,2):$30}$P^{\circ P}$, {(1,3):$45}$P^{\circ P}$, {(1,4):$55}$P^{\circ P}$, {(1,1), (2,2):$70}, {(2,2):$60}, {(2,2), (3,1):$75}, {(2,1),(3,2):$60}, {(3,1):$30} $P^{\circ P}$ indicates singleton vectors.
1	{(1,1), (2,2):$70}, {(1,1):$20}$P^{\circ P}$, {(1,2):$30}$P^{\circ P}$, {(1,3):$45}$P^{\circ P}$
2	{(2,2):$60}, {(2,2), (3,1):$75}, {(2,1),(3,2):$60}
3	{(3,1):$30}

Basket 1 : {(1,1), (2,2):$70}, {(1,1):$20}

Basket 2 : {(2,1),(3,2):$60}, {(3,1):$30}

Bids in basket 1 are removed keeping basket 2 intact. The pairs of (items, units) freed are (1, 2) and (2, 2). However including the one unit of item 1 which remain unallocated after the initial complete allocation, the items and units subjected to search are (1,3) and (2,2). The corresponding participative bids are {(1,1):$20}, {(1,2):$30}, {(1,3):$45}, {(1,4):$55}, {(1,1), (2,2):$70}.

DFBB search on these lead to the optimum solution revenue of $105, the bids being {(1,3):$45}, {(2,2):$60}. This solution added to basket 2 gives the complete neighborhood solution as {(1,3):$45}, {(2,2):$60}, {(2,1),(3,2):$60}, {(3,1):$30} with revenue 195, an improvement over the present solution and hence this becomes the present solution. Now the bids in basket 2 are removed. The items and units now subjected to search are (2,1) and (3,3). The corresponding participative bids are {(1,1):$20}, {(1,2):$30}, {(1,4):$55}, {(1,1), (2,2):$70}, {(2,2),(3,1):$75}, {(2,1),(3,2):$60}, {(3,1):$30}. The only feasible solution for search in this basket is the presently existing set of bids in it so that there is no improvement. Hence after iteration 1, there was an improvement and the solution becomes {(1,3):$45}, {(2,2):$60}, {(2,1),(3,2):$60}, {(3,1):$30} with revenue 195. This process is continued until a specified number of non improving iterations.

Role of β in LSMU(β)

As β increases, the sub-problem that needs to be solved for finding a neighborhood solution reduces in size and the algorithm runs faster. The disadvantage is that since the multiple sub-problems need to be additive, the DFBB gets performed on the sub-problem while adhering to the constraint that the units in a complete allocation do not exceed that available for auction. More the number of partitions, fewer would be the units free for search, higher will be the probability of having non-participative bids and more would be the compromising of solution quality. The strength of LSMU(β) lies in finding a suitable value of β that exhibits its superior performance.

Experimental Setup

A large number of experiments were conducted to explore the efficacy of LSMU(β) in terms of computational time and quality of solution and also to find an appropriate choice of β. Since available literature [Kevin, 2000b] deals with problem instances mostly drawn from uniform and decay distributions, we have experimented with both distributions for better insight.

a) Uniform Distribution: The test suite was generated using the uniform distribution. Given the values of number of good and number of bids, for each problem instance, number of units available for every item, items present in every bid and their number of units, and cost of every bid were generated using uniform distribution:

b) Decay distribution: This distribution is similar to the one given by Kevin et al [Kevin, 2000b] which generates bids with fewer items irrespective of the total number of items and that they are computationally harder to solve compared to drawing goods from a uniform distribution. The number of units for each good in a bid is also fewer. The uniform distribution is used for generating the number of units available for every item, the identity of items in a bid and the cost of every bid. The decay distribution with parameter $\lambda = 5$ is

used for generating the number of items in a bid and the number of units for the items in the bid.

Size of Problem Instances: To evaluate the empirical performance of LSMU (β), the number of partitions β was varied from 1 to 6, and their effect on percentage cpu time saved and percentage revenue obtained vis-à-vis LSMU(1) were observed; the solution obtained by LSMU(1) using DFBB is bound to be optimal. The problem size is denoted using the nomenclature u(items,bids) and d(items,bids) for the uniform and decay distributions respectively and given in Table 2. The maximum price (cost) for any bid is set at 2000. The parameter T for the number of successive iterations without any improvement was kept at 100.

We also study the effect of partitioning on a few very large problem sizes. The large problem instances were isolated from Table 5.1 based on the time LSMU(1) takes to solve them. We select 10 instances that take largest CPU time. The time is definitively more than 5 hours and preferably much larger. These instances have been identified in table 6.1 reported in Section 6.2.

Table 2. Experimental test data specifications

Problem	Number of goods	Number of Bids	Maxunit	Number of test runs
u(20,500)	20	500	8	50
u(20,600)	20	600	8	50
u(20,700)	20	700	8	50
u(20,800)	20	800	8	50
u(20,900)	20	900	8	25
u(40,500)	40	500	8	50
u(40,600)	40	600	8	50
u(40,700)	40	700	8	25
u(60,400)	60	400	8	25
d(10,100)	10	100	5	20
d(10,125)	10	125	5	20
d(10,150)	10	150	5	10
d(15,100)	15	100	5	20
d(15,125)	15	125	5	20
d(20,100)	20	100	3	20
d(20,125)	20	125	3	20
d(25,100)	25	100	3	20
d(25,125)	25	125	3	20

Maxunit = Maximum number of Units available for any good with the auctioneer.

Empirical Results

In this section we report our experimental findings under different subsections.

Effect on Revenue proximity to optimal

The proximity of revenue obtained to the optimal decreases in general as partitioning increases. However, in absolute terms the drop in revenue is not much. The revenue is always more than 98% irrespective of the partition sizes up to 6. For 4 partitions, the revenue obtained is always closer to 99.5 %.

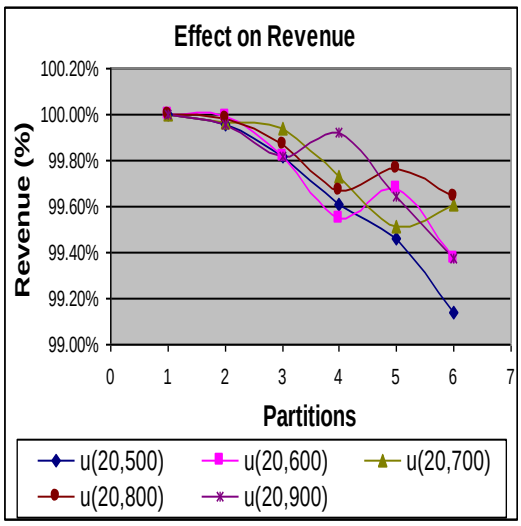


Figure 1. Effect on Revenue

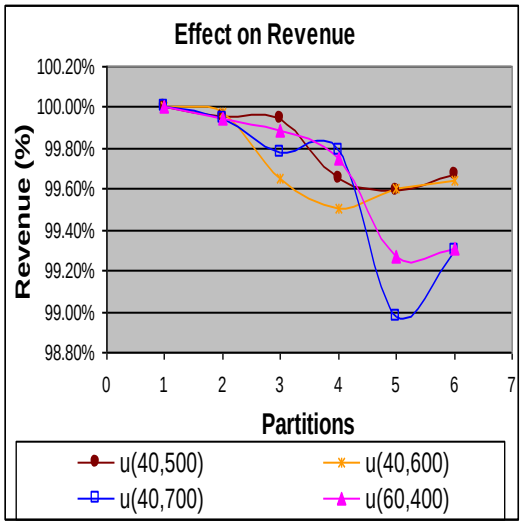


Figure 2. Effect on Revenue

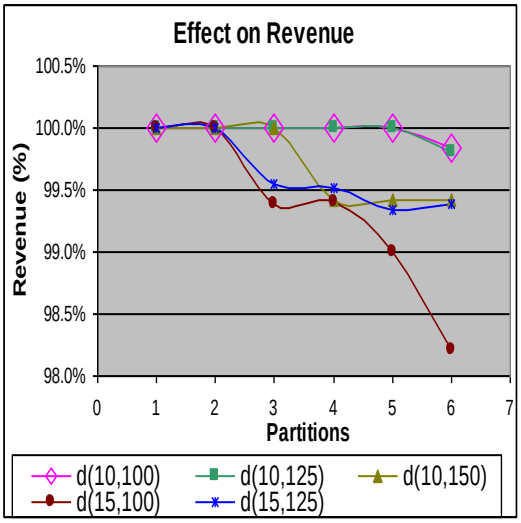


Figure 3. Effect on Revenue

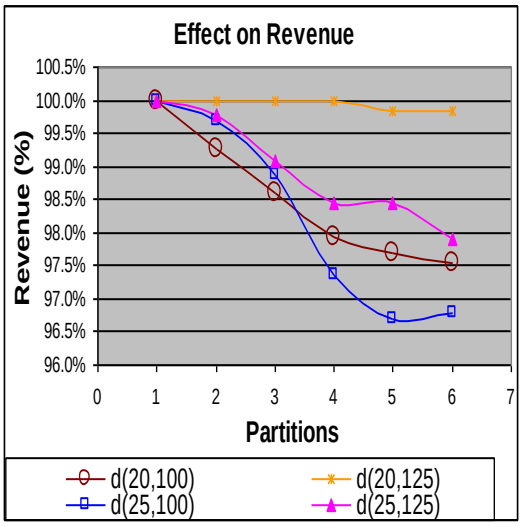


Figure 4. Effect on Revenue

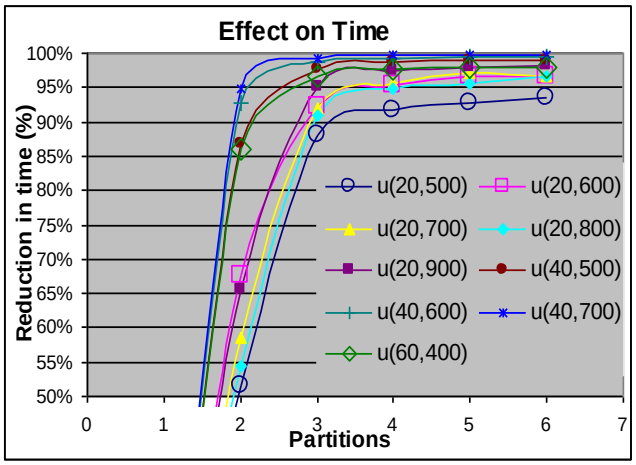


Figure 6. Effect on CPU time

Effect on saving in CPU time

The average percentage savings in time with partitions for the uniform distribution is shown in Figure 5. The average percentage savings in time shows a non-decreasing trend originating at partition 1 with 0% savings in time going up to maximum of 99.74%. The average percentage savings in time is always above 87% at

and after partition 3 for all problem sizes and more than 90% for most of them. The interesting feature about these plots is that the percent savings for relatively larger problems namely u(40,500), u(40,600), u(40,700) and u(60,400) outflank the relatively smaller problems of 20 item size. Partition size of 3 and 4 appear very promising.

For the decay distribution, depicted in Figure 6 and Figure 7, the trends are similar to the uniform distribution. For most of the problem sets, the average

Figure 6: Effect on CPU time

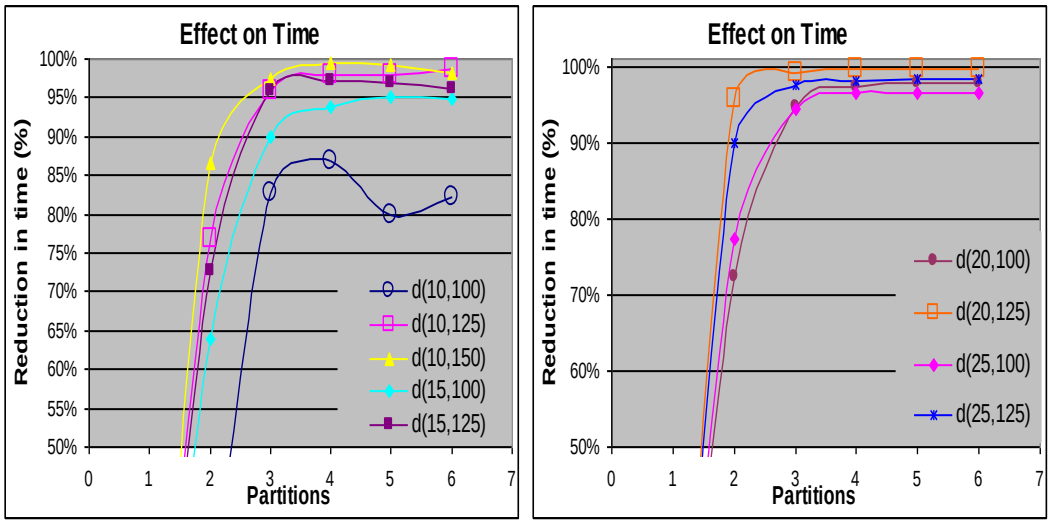
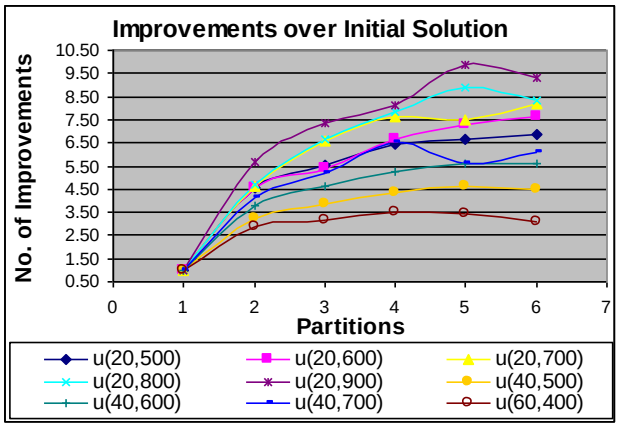


Figure 7: Effect on CPU time

percentage savings in time is more than 90% from partition 3 onwards. Looking at the results of revenue efficacy and percentage savings in time, it appears that a partition of 3 to 4 is justified for solving these problem sizes.

Fig 8: Improvements over initial solution



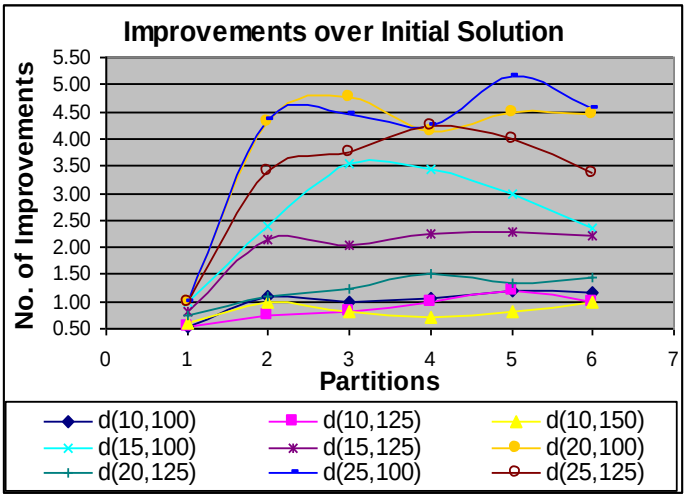
Improvements over Initial Solution

The LSMU (β) algorithm when β >1 continues to attempt improvisation of the revenue till a specified number of non improving iterations have occurred. For the present tests we fixed these iterations at 100. As observed revenue efficacy of the results look very impressive. With increase in partitioning, the total number of improvements also increases. This is expected as with increase in partitions, each partition is of a smaller size and so

the scope for perturbation of bids and consequently on revenue is less. For the uniform distribution (Figure 8) the effect is varied across different problem sizes.

Also the rate of increase appears to be higher for smaller problems than the larger ones.

Fig 9: Improvements over initial solution



.

For the decay distribution (Figure 9) the trend is the same that of increase in improvements with partitions though there is a marked variation with problem sizes. However unlike the uniform distribution for a number of problem sizes specially the relatively smaller ones like d(10,100), d(10,125) and d(10,150), for a number of partitions, the average improvements is less than 1. The corresponding revenue quality results are very close to 100%. The improvements with problems of the decay distribution are relatively lesser than with uniform distribution. Unlike the uniform distribution, the higher improvements are observed with the relatively larger problem sizes of d(20,100), d(25,100) and d(25,125). Also for these problems, on an average improvement at partition 1 is 1 unlike the smaller ones where it is around 0.5. The trends show that the main rise in average improvements happens from partition 1 to partition 2 after which the slope is relatively flatter.

Large data instances

Table 3 identifies the large data instances analyzed for our results. The CPU time LSMU(1) takes to solve these instances varies from 19,400 – 2,00,400 secs. Results for the large instances show that even for instances which take a huge amount of CPU time to solve, LSMU(β) solves them extremely fast by increasing the partitions without any significant deterioration in percentage revenue obtained. The savings in time increases drastically with partitioning. The effect can be gauged from the very first and the most difficult of the problem i.e 15 from the u(40,700) problem set. The optimal for this took 200414.8 secs to solve. However with subsequent partitions, the timings reduced to 804 secs at partition 2, 167.5 secs at partition 3 to finally just 11.5 secs at partition 6. This contrasted with the percentage revenue obtained, shows that for the partitions 2, 3 and 6, the revenue obtained is optimal and at partitions 5 and 6, it is as high as 99.3 %. The results for the remaining large problems are similar and tabulated in Table 6.2. In six out of these 10 instances, the revenue was optimal upto partition 4 and in three instances it was optimal in all the partitions. These results suggest that it might be advantageous to use multi partitions to solve very large problems which otherwise might take some hours of CPU time without bothering much about loss of revenue. In the process it also throws up certain observations and questions on the possible choice of β for solving large instances.

Table 3. Large Instances from Table 6.1

Problem Set	Problem Instance	CPU time by LSMU(1) in secs
u(40,700)	15	200414.8
u(20,900)	6	54734.74
u(40,600)	48	43614.13
u(40,700)	2	36022.24
u(40,600)	12	33049.6
u(40,600)	34	32415.05
u(40,700)	9	29136.29
u(40,600)	20	22028.71
u(40,600)	3	21076.74
u(40,700)	17	19482.66

Table 4 Empirical Results for Large Instances

Problem Set, Instance, Optimal time in secs		Partitions					
		1	2	3	4	5	6
u(40,700), 15, 200414.8 secs	% of optimal	100.0	100.0	100.0	100.0	99.3	99.3
	% saving in time	0.00	99.60	99.92	99.99	99.99	99.99
u(20,900),6, 54734.74 secs	% of optimal	100.0	100.0	100.0	100.0	98.26	98.58
	% saving in time	0.00	95.51	99.81	99.94	99.93	99.99
u(40,600),48, 43614.13 secs	% of optimal	100.0	100.0	100.0	100.0	100.0	100.0
	% saving in time	0.00	98.82	99.75	99.96	99.95	99.98
u(40,700),2, 36022.24 secs	% of optimal	100.0	100.0	100.0	100.0	100.0	100.0
	% saving in time	0.00	95.08	99.61	99.81	99.93	99.94
u(40,600),12, 33049.6 secs	% of optimal	100.0	100.0	97.09	97.09	97.09	97.09
	% saving in time	0.00	98.26	99.90	99.96	99.98	99.98
u(40,600),34, 32415.05 secs	% of optimal	100.0	100.0	99.27	99.35	99.27	100.0
	% saving in time	0.00	97.53	99.93	99.94	99.98	99.98
u(40,700),9,	% of optimal	100.0	100.0	100.0	100.0	95.75	100.0

29136.29 secs	% saving in time	0.00	97.86	99.73	99.88	99.95	99.91
u(40,600),20, 22028.71 secs	% of optimal	100.0	100.0	94.89	94.89	100.0	99.81
	% saving in time	0.00	98.72	99.90	99.96	99.96	99.97
u(40,600),3, 21076.74 secs	% of optimal	100.0	100.0	99.95	99.95	99.95	99.95
	% saving in time	0.00	97.47	99.80	99.91	99.95	99.95
u(40,700),17, 19482.66 secs	% of optimal	100.0	100.0	100.0	100.0	100.0	100.0
	% saving in time	0.00	98.25	99.81	99.93	99.94	99.97

Choice of β in LSMU(β)

The two useful observations from our experiments are:

(a) As the partitions are increased, the computation time reduces drastically but quality of solution does not degrade at that rate – the degradation rate is far slower, a seemingly counter-intuitive result. This allows room for LSMU(β) in finding acceptable solution within affordable time.

(b) In general, it appears difficult to suggest a possible choice of β. The longer LSMU(1) takes to solve an instance, higher is the expected value of β for solving the instance within reasonable CPU time. One possible way to choose the β is to start from a large value of β determined by the number of bids in the initial solution, and gradually decrease its value to get a solution with near optimal solution quality.

(c) Interestingly, the experimental results indicate 3 or 4 partition would be ideal even for instances taking longer time with LSMU(1) - significant computational savings can be had at little compromise on the quality of solution. For uniform distribution, the average proximity of results to optimal is more than 99.5% and the percentage savings in time varies from 87% to 99% for partition 3 and 92% to 99% for partition 4. For the decay distribution, the average proximity of results to optimal is more than 96.5% and the average percentage saving in CPU time varied from 83% to 99.7% for partition 3 and 87% to as high as 99.8% for partition 4. This finding suggests that based on the number of bids in the initial solution, one can possibly start with the value of β set to either 3 or 4. If running time goes beyond a certain acceptable limit, one can start from a higher value and subsequently reduce β as described in (b).

Conclusion

Multi-unit winner determination problem in combinatorial auction is a notoriously hard optimization problem with direct practical application to electronic commerce.. In this paper we have proposed a parameterized

stochastic local search algorithm LSMU(β) to solve the problem. The number of partitions, β, can be used to compromise between solution quality and time of computation. The most useful and interesting finding is that the choice of β is not open-ended and the experiments indicate a value equal to either 3 or 4. At this choice of β, in comparison to finding optimal solutions, the saving in computation of LSMU(β) is significant whereas the degradation of the quality of solution is insignificant. These findings make LSMU(β) a promising algorithm for solving large MWDP instances which would be difficult to solve otherwise.

This research throws up a few interesting questions which can be a scope of further research.

- What is the effect of initial solution on the final solution in such local search algorithms?
- The algorithm LSMU(β) determine the neighborhood for search in a random fashion by removal of a predetermined number of bids. Is there a much more intelligent way of selection, which would improve upon the results?
- Can the learning acquired over multiple partitioning be used to get a better quality solution or a very near to optimal solution? The bids which are common in the solution for different partitions appear to have a higher chance of being in the optimal solution. These bids may be kept fixed and the remaining bids may be searched optimally using DFBB. With this strategy there seems to be a possibility of striking optimal frequently and that too very fast.

References

Adomavicius G. and A. Gupta. 2005.. Toward Comprehensive Real-Time Bidder Support in Iterative Combinatorial Auctions. *Information Systems Research*, 16(2): 169 - 185.

Ausubel. L. M. and P. R. Milgrom. 2002. Ascending auctions with package bidding. Frontiers of Theoretical Economics.1(1):1-42.

Clarke, Edward. H. 1971. Multiple Pricing of Public Goods. Public Choice, 11: 17–33.

Cramton Peter, Shoham Yoav and Steinberg Richard. 2006. Combinatorial Auctions. Editors. Massachusetts. MIT Press.

Fujishima, Y., K. Leyton-Brown, and Y. Shoham. 1999. Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches, In Proceedings of the International Joint Conference on Artificial intelligence (IJCAI-99). Stockholm: 548-553.

Garey, Michael R. and David S. Johnson. 1979. Computers and Intractability. New York. W.H. Freeman and Company.

Gonen, Rica and Daniel Lehmann. 2000. Optimal Solutions for Multi-Unit Combinatorial Auctions: Branch and Bound Heuristics. In Proceedings of the 2nd ACM conference on Electronic commerce (EC-00). Minneapolis. Minnesota. US. Oct: 13–20.

Gonen, Rica and Daniel Lehmann. 2002. Linear Programming helps solving large multi-unit combinatorial auctions. CoRR cs.GT/0202016.

Ghebreamiak, Kidane Asrat and Arne Andersson. 2002. Caching in multi-unit combinatorial auctions. In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems. Bologna. Italy: 164–165.

Groves, Theodore. 1973. Incentives in Teams. Econometrica. 41: 617–631.

Hoos, Holger H. and Craig Boutilier. 2000. Solving Combinatorial Auctions using Stochastic Local Search. In Proceedings of the National Conference on Artificial Intelligence **(**AAAI-00): 22-29.

Hoos, Holger H. and Thomas Stutzle. 2004. Stochastic Local Search: Foundations and Applications. San Francisco. Morgan Kaufmann Publishers Inc. US.

Kevin Leyton-Brown, Mark Pearson and Yoav Shoham. 2000a. Towards a Universal Test Suite for Combinatorial Auction Algorithms. In Proceedings of the ACM Conference on Electronic Commerce (EC-00): 66-76.

TKevin Leyton Brown, Yoav Shoham and Moshe Tennenholtz. 2000b. An Algorithm for Multi Unit Combinatorial Auctions.T In Proceedings of the National Conference on Artificial Intelligence (AAAI-00). Austin. TX. US: 56-61.

Koksalan M, R Leskela, H. Wallenius and J. Wallenius. 2009, Improving Efficiency in Multiple-Unit Combinatorial Auctions: Bundling Bids from Multiple Bidders, *Decision Support Systems* 48: 103 - 111.

Lehmann, D., Mueller, R., and Sandholm, T. 2006. The Winner Determination Problem. Chapter 12 of the book Combinatorial Auctions. Cramton, Shoham, and Steinberg, eds. MIT Press.

Muller Rudolf. 2006. Tractable Cases of the Winner Determination Problem. Chapter 13 of the book Combinatorial Auctions. Cramton, Shoham, and Steinberg, eds. MIT Press.

Parkes, D. C. 2006. Iterative Combinatorial Auctions, Chapter 2 of the book Combinatorial Auctions. Peter Cramton, Yoav Shoham and Richard Steinberg. eds.MIT Press.

Parkes. D. C. 1999. iBundle: An efficient ascending price bundle auction. First ACM Conference on Electronic Commerce. Nov: 148-157.

Milgrom, Paul R. and Weber, R. J. 1982. A Theory of Auctions and Competitive Bidding. Econometrica. 50(5): 1089–1182.

Raj Jog Singh and Anup K. Sen. 2005. Winner Determination Problem: Experimenting with Local Search. In Proceedings of the Workshop on Information Technologies and Systems (WITS 2005). Las Vegas. US. Dec: 213-218.

Rothkopf, Michael H., Alexander Pekec and Ronald M Harstad.. 1998. Computationally Manageable Combinatorial Auctions. Management Science. 44(8):1131–1147.

Sandholm, Tuomas W., Subhash Suri, Andrew Gilpin and David Levine. 2001. CABOB: A Fast Optimal Algorithm for Combinatorial Auctions. In Proceedings of the International Joint conference on Artificial intelligence (IJCAI-01). Seattle. WA. US: 1102-1108.

Sandholm, T. 2006. Optimal Winner Determination Algorithms. Chapter 14 of the book Combinatorial Auctions. Cramton, Shoham, and Steinberg, eds. MIT Press.

Sandholm, Tuomas W. 2002. Algorithm for Optimal Winner Determination in Combinatorial Auctions. Artificial Intelligence. 135: 1–54.

Sandholm, Tuomas W. and S. Suri. 2003. BOB: Improved Winner Determination in Combinatorial Auctions and generalizations. Artificial Intelligence.145: 33-58.

Vickrey, William. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. Journal of Finance. 16: 8-37.

Wilson R. 1969. Competitive bidding with disparate information. Management Science. 15: 446- 448.

Xia, M., Stallaert, J. and Whinston, A.B. 2005. Solving the combinatorial double auction problem. European Journal of Operational Resesarch. 164: 239-251.